Alocação de memória e listas encadeadas (supor ambiente Linux)

Analise as afirmações abaixo e julgue se são verdadeiras (V) ou falsas (F):

```
typedef struct _no {
    int              info;
    struct _no      *prox;
    struct _no      *anterior;
} no;
int       vet[10], *r;
no        *p, q;
```

a) ( ) No exemplo acima, a variável `r` (apontador para um tipo *int*) ocupa menos espaço na memória do que a variável `p` (apontador para o tipo ***no***)

b) ( ) Se a variável `q` é do tipo ***no***, `q` ocupa um espaço para um inteiro e dois espaços para apontadores, totalizando 12 bytes numa arquitetura de 32 bits (Kernel 32 bits, GCC 32 bits) e 24 bytes numa arquitetura de 64bits (Kernel 64 bits, GCC 64 bits)

c) ( ) Se `p` é um apontador para o tipo ***no***, então `p` ocupa um espaço para um inteiro e dois espaços para apontadores, totalizando 12 bytes numa arquitetura de 32 bits (Kernel 32 bits, GCC 32 bits) e 24 bytes numa arquitetura de 64bits (Kernel 64 bits, GCC 64 bits)

d) ( ) Considerando uma arquitetura/compilador de 32 bits, o vetor `vet` ocupa 40 bytes (assumindo um `int` de 4 bytes), enquanto uma lista encadeada com o tipo ***no*** ocupará 120 bytes para os mesmos 10 elementos

e) ( ) Listas encadeadas gastam mais espaço do que vetores, porém são mais flexíveis, podendo crescer ou diminuir em função da necessidade do programador

f) ( ) Pode-se dizer que a inserção de um novo elemento num vetor ordenado envolve pelo menos uma das seguintes estratégias: *(i)* encontrar a posição correta do vetor para inserção e criar o espaço (movimentando os elementos subsequentes), com cuidado para não ultrapassar o limite do vetor; *(ii)* inserir o elemento no final do vetor e aplicar algum algoritmo de ordenação (*bubble sort*, por exemplo)

g) ( ) Pode-se dizer que a inserção de um novo elemento numa lista encadeada, para uma quantidade grande de elementos, em média exige menos processamento do que vetores de mesmo tamanho. No caso de uma lista, bastaria criar o elemento e encontrar a posição correta de inclusão na lista, enquanto nos vetores as exigências envolvem o que foi citado no item anterior

h) ( ) Pode-se dizer que listas encadeadas e suas derivações (duplamente encadeada, circular, ...) são estruturas de dados mais complicadas de manipular e consomem mais memória do que vetores; no entanto, são uma alternativa interessante para uma série de problemas computacionais.